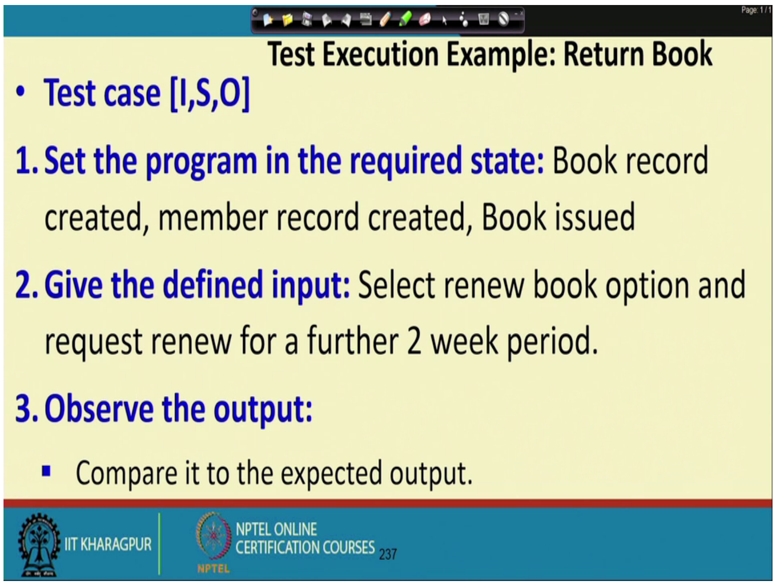
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 45
Basic concepts in Testing-III

In the last lecture we had discussed about some very Basic concept and Testing. We were discussing about the test cases test data. And, we had said that for every software a set of test cases are designed carefully and this is known as the test suite.

(Refer Slide Time: 00:52)



Test Execution Example: Return Book

- Test case [I,S,O]

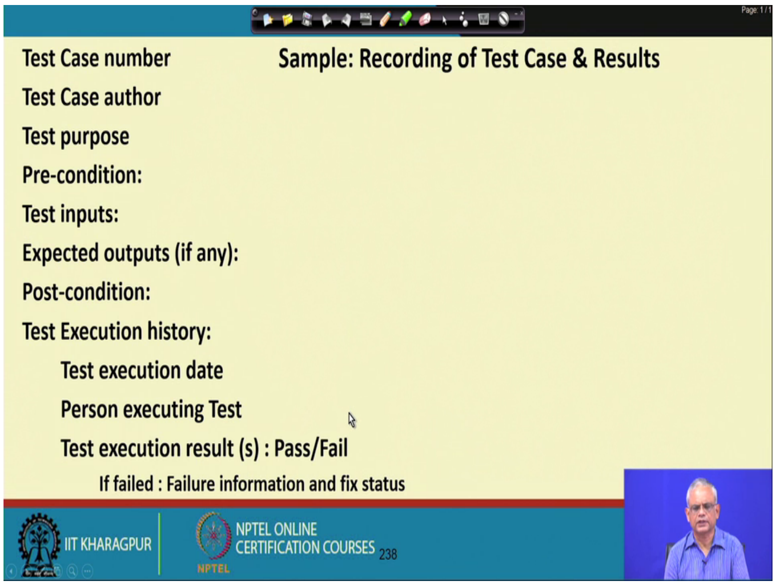
1. Set the program in the required state: Book record created, member record created, Book issued
2. Give the defined input: Select renew book option and request renew for a further 2 week period.
3. Observe the output:
   - Compare it to the expected output.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
237

And for each test case the system we execute the software using the test data. And each test case is typically contains 3 parts. The test data the state at which the system should be there when the test input is given another expected output. So, to execute the software with a test case, we first put the software in the required state S.

For example, some book record might have to be created; some member record might have to created books issued for a library software. And, once it is there in the required state then will be the input and then observe the output. And, we note this the observed we observe the output and note our conclusion on a test report let see how a test report is organized?

(Refer Slide Time: 02:02)



Sample: Recording of Test Case & Results

Test Case number

Test Case author

Test purpose

Pre-condition:

Test inputs:

Expected outputs (if any):

Post-condition:

Test Execution history:

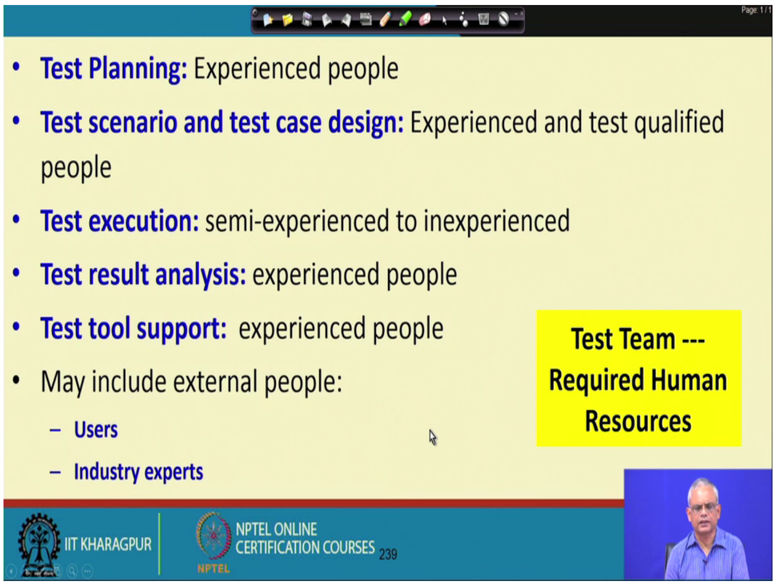
- Test execution date
- Person executing Test
- Test execution result (s) : Pass/Fail
  - If failed : Failure information and fix status

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
238

Typically, each test case has a number it becomes easier to see that which test case got executed, because once the system fails the developer's needs to reproduce the bug. So, test case can uniquely identify, which test case they should execute, who had written the test case the author? The purpose of the test case like, which functionality or performance etcetera it is designed to test, the test data or the test input the expected output ok. Pre-condition is the state at which the system should be there, test input expected output and the state at which the system should be left, after the test execution completes.

And, once the test output is produced the testers observe the output and then they write their analysis of the ex observed output. They write the test, execution date person executing the test, they observe the pass fail whether it has passed the test case or it failed the test case. And, if failed they write what exactly happened? What was observed during the failure? They did crash they did produce a wrong result; they did go into an infinite loop and so on. And then there is another field because the developers need to fix each of the failure cases of test cases. So, there is a fix status field where the developers can record, whether they are fixed it or not yet fixed.

(Refer Slide Time: 04:09)



- **Test Planning:** Experienced people
- **Test scenario and test case design:** Experienced and test qualified people
- **Test execution:** semi-experienced to inexperienced
- **Test result analysis:** experienced people
- **Test tool support:** experienced people
- May include external people:
  - Users
  - Industry experts

Test Team --- Required Human Resources

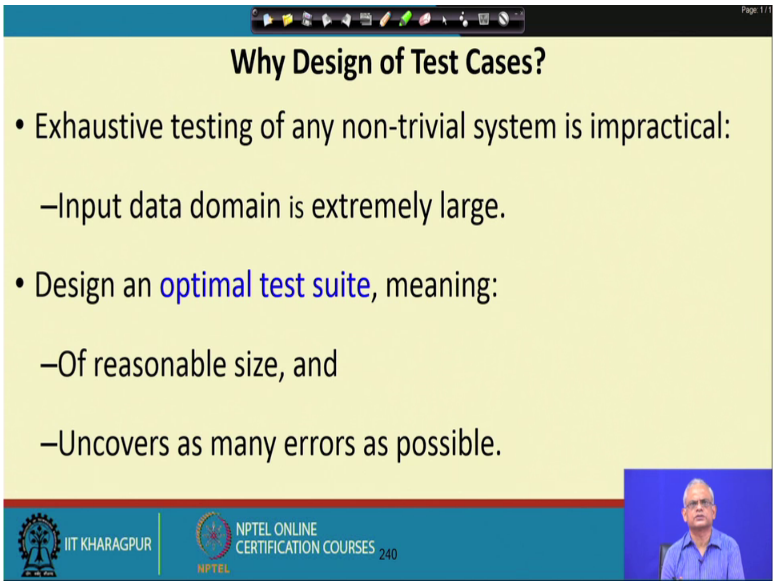
IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 239

Now, let see the composition of the testing, that is when a organization develop software; what types of testers does it need? You saw that there are many activities during testing, the different activities of testing are test planning, test scenario and test case design, test scenario design test case design we will see, what is the difference between a test scenario and test case? But, roughly speaking a test scenario is a abstract description of a test case. For example, we might say that all parts should be covered or a specific part needs to be covered.

So, that said test scenario that has specific path in the program needs to be executed. Whereas, the test case is more concrete then the test scenario in the sense that we provide the test data input data for which that specific path will be covered test execution, test result analysis, test tool uses and so on. So, there are many activities during testing and let see what are the kind of human resource we need for the testing, what test planning? This is done by experienced people, test scenario design and test case design this also experienced qualified people.

Test execution can be done by a semi experienced to inexperienced persons, depending on the software that is being tested. Test result analysis this is the experienced people and test tool users the experienced people and also testing several other people participate like users industry experts and so on. So, the composition of the testing is of various types of personal, because different activities in testing require different levels of expertise.

(Refer Slide Time: 06:47)



Why Design of Test Cases?

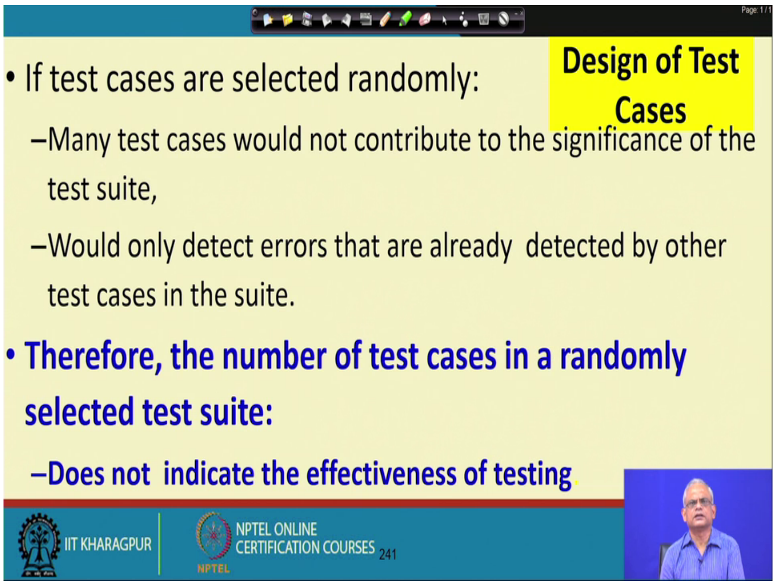
- Exhaustive testing of any non-trivial system is impractical:
  - –Input data domain is extremely large.
- Design an optimal test suite, meaning:
  - –Of reasonable size, and
  - –Uncovers as many errors as possible.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
240

But, let us now answer a very fundamental question, why do you need to design test cases? Cannot we just test it will all possible inputs, because if we could test with all possible inputs, then we can be see your that the software is working fine, but why I go about only designing few test cases and then execute those. The answer to this question is that, even for a very small trivial almost trivial software. If we want to test it exhaustively it is extremely large number of test cases trillions of test cases can be designed for exhaustive testing and that input data domain is extremely large, and it will take thousands or millions of year to even test a simple program.

We will see through an example that why this is so? We will take a simple program and see how many test cases are required to test it for all possible test inputs. Since, it is impractical to test with all possible inputs; we need to develop a optimal test suite. This optimal test suite should be of reasonable size and it should be able to cover uncover the largest or a maximal number of errors.

(Refer Slide Time: 08:46)



Design of Test Cases

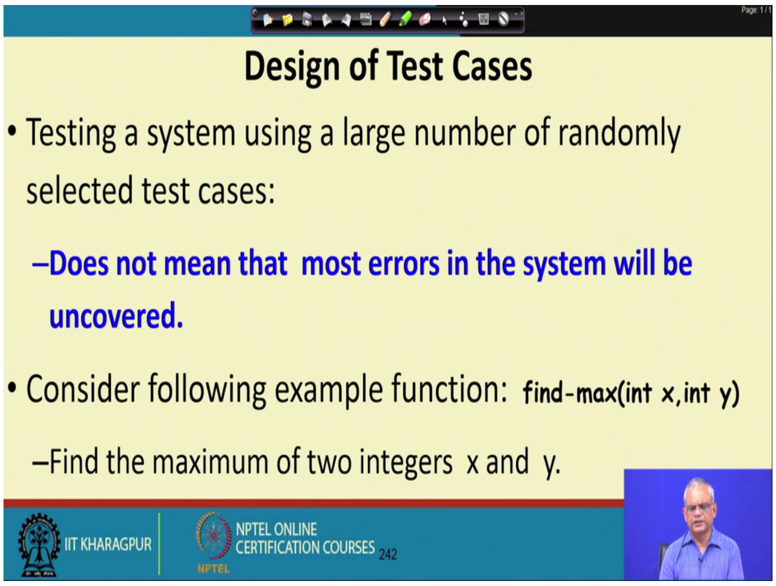
- If test cases are selected randomly:
  - Many test cases would not contribute to the significance of the test suite,
  - Would only detect errors that are already detected by other test cases in the suite.
- **Therefore, the number of test cases in a randomly selected test suite:**
  - **Does not indicate the effectiveness of testing**

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
241

Also, we cannot give random data, because random data has a problem that we might give the same data again or we might give a data which detects similar problem, similar bug, then the test cases that were given earlier, we will see through an example. So, even though we might test a software using 1000 test cases, a very small function let say we tested with 1000 test cases, but still if we are doing it in random testing, we cannot be see your that it has been tested well.

Because most of the test cases that have been generated randomly, may be detecting the already detected bugs in that test suite. So, if the test cases are generated randomly, then claiming that we tested it using a large number of test cases, does not indicate truly the effectiveness of testing we might have tested with million test cases, but if it just random test cases, then it does not mean that most of the bugs have been eliminated might just have detected very small number of bugs if at all.

(Refer Slide Time: 10:25)



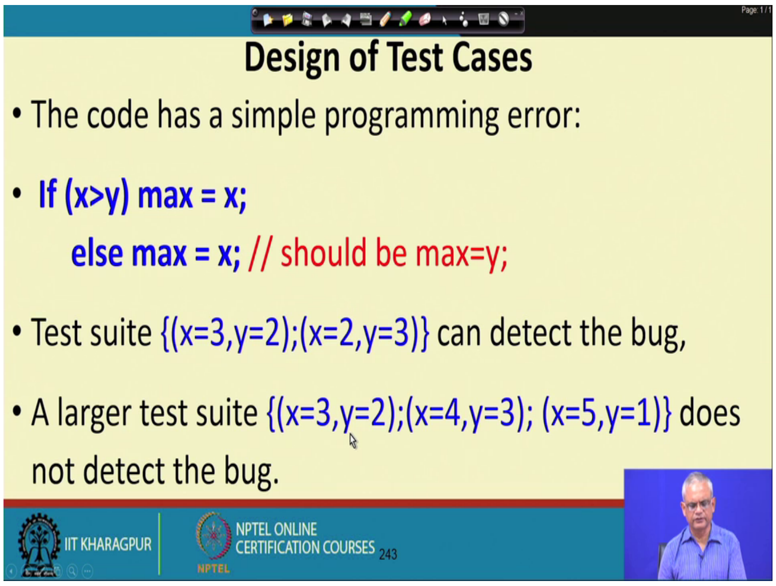
Design of Test Cases

- Testing a system using a large number of randomly selected test cases:
  - Does not mean that most errors in the system will be uncovered.
- Consider following example function: find-max(int x,int y)
  - Find the maximum of two integers x and y.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
242

Now, let see how to design test cases? So, testing using random test cases does not mean that there system is tested well; most of the problems have been uncovered. Let us convince ourselves with small example. Let say we have a function name of the function is find max and the parameters are two integer parameter x and y and the function let say is a trivial function, which just written the larger of the two the code is just 2 lines.

(Refer Slide Time: 11:19)



Design of Test Cases

- The code has a simple programming error:

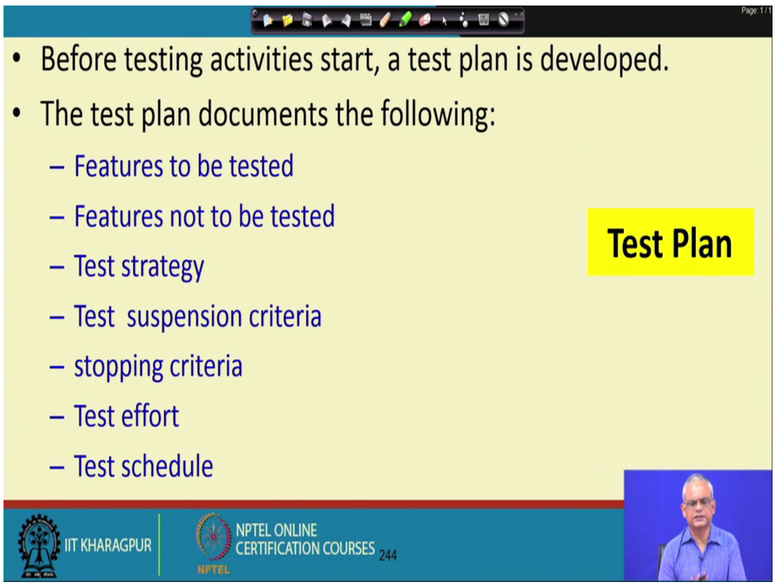
-
  ```
  If (x>y) max = x;
     else max = x; // should be max=y;
  ```
- Test suite {(x=3,y=2);(x=2,y=3)} can detect the bug,
- A larger test suite {(x=3,y=2);(x=4,y=3); (x=5,y=1)} does not detect the bug.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
243

But, then let say in this 2 line code we have made a mistake. If x greater than y max equal to x else max equal to it should have been y, but by mistake programmer has written max equal to x. Now, if we give 2 test cases for the test cases only represented the test data omitted the other for conciseness and understand ability. Now, the test should where we test it with x is equal to 3 y equal to 2 that is the first statement is executed.

And, x is equal to 2 y equal to 3 that is a second statement is executed we can detect the bug. Because, if x greater than y then we should report that the max is x. Otherwise we should report max is equal to y, but then here the bug will get exposed, but let say we have a larger test suite where only the first statement is always executed. Even, if we have a thousand test cases like this, where x is larger than y and only the first statement is executed and the bug that is there in the second statement is will not be detected.

(Refer Slide Time: 13:01)



- Before testing activities start, a test plan is developed.
- The test plan documents the following:
  - Features to be tested
  - Features not to be tested
  - Test strategy
  - Test suspension criteria
  - stopping criteria
  - Test effort
  - Test schedule

Test Plan

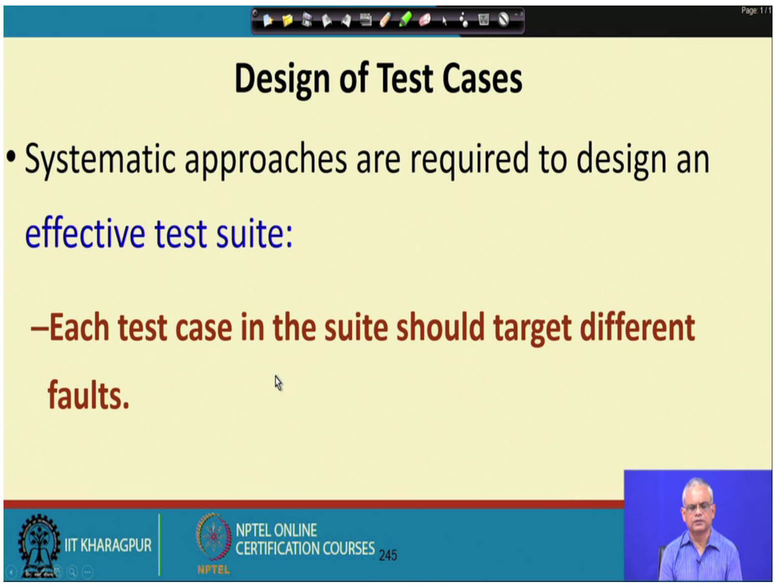
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
244

So, that example should convince us that, if we have a large number of random test cases that does not mean that we have tested the software well. Now, let see what exactly the test plan, because before testing starts the test plan is prepared. The contents of the test plan are that for a certain testing to be carried out, what are the features to be tested?

That is what features have been already completed and those need to be tested now, what features not to be tested? May be those features are still undergoing development and need not be tested, the test strategies that are deployed we will see that there are many strategies. The black box and white box and also there are several strategies of black box testing and several strategies for white box testing. Test suspension criteria, that is if a fatal error let say crash is observed then we may not be able to run the other test cases.

And then we must suspend the test case sorry testing and then ask the developers to fix the problem before the other testing can be done, the test suspension criteria says that when under which situation, the test cases can be the test execution can be suspended and the developers ask to fix the bugs before a testing is resumed. Stopping criteria this says that as the testing goes on progressively less box are detected and how long to test? The test effort is what is the plan test effort that is required how many people, testers how many testers, for how many days and so on? The test schedule is about when, what type of testing will be done?

(Refer Slide Time: 15:30)

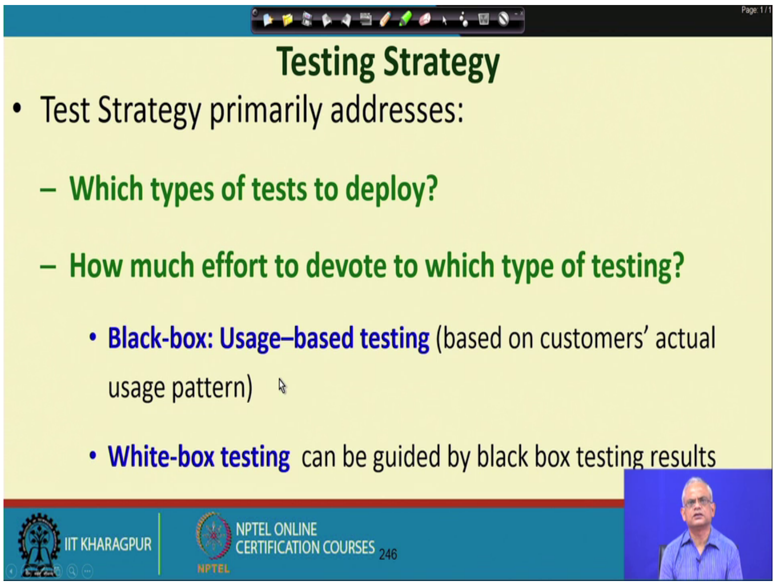


Design of Test Cases

- Systematic approaches are required to design an effective test suite:
  - Each test case in the suite should target different faults.

Now, let see how to go about designing test cases? One very important concept here is that when we design test cases each of our test case should target to detect different types of faults or bugs. No 2 test cases should actually target the same bug, because that will only increase the number of test cases without reducing the bugs in the code.

(Refer Slide Time: 16:08)



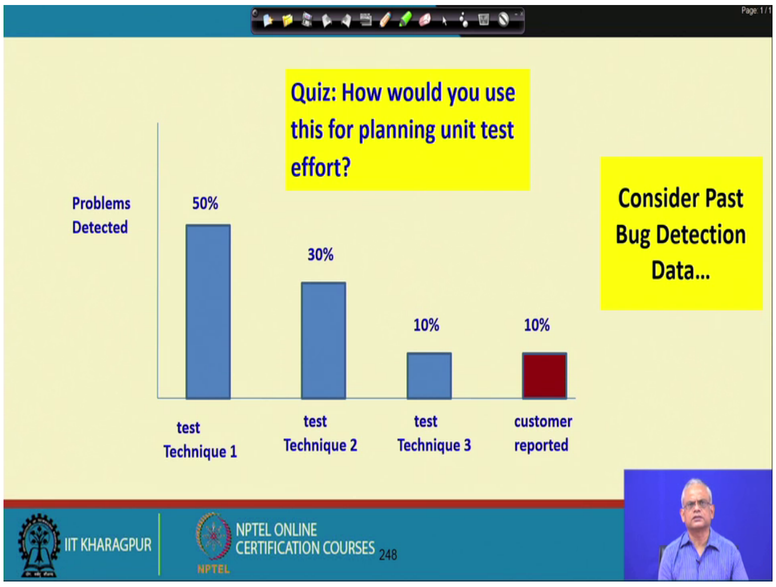
Testing Strategy

- Test Strategy primarily addresses:
  - Which types of tests to deploy?
  - How much effort to devote to which type of testing?
    - **Black-box: Usage–based testing** (based on customers' actual usage pattern)
    - **White-box testing** can be guided by black box testing results

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 246

Now, let see the test strategy. So, far we have been saying that there are many test strategy and these can be considered as bug filters and so on. Now, let see at very overview level, what are the test strategies?

(Refer Slide Time: 16:48)



Quiz: How would you use this for planning unit test effort?

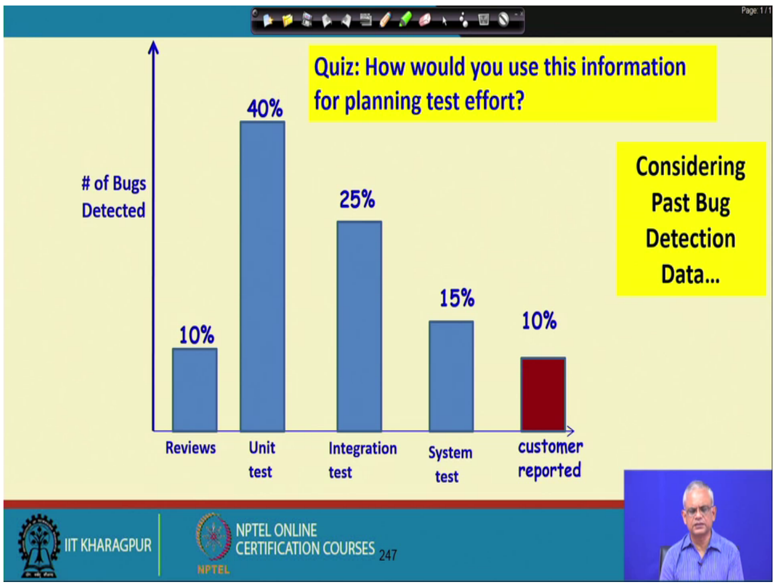
Consider Past Bug Detection Data...

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES 248

The test strategy is basically type of test to be deployed; there are many types of test strategy as we will see subsequently. The white box and black box is a graph characterization of the test strategies, but then there are many types of white box strategies and many types of black box strategy, but given that there are dozens of strategies to be deployed for testing. The testers have to plan how much time should be devoted for which type of testing? We will see how to plan for the time? But, then for the 2 types of testing; the black box these are also called as the usage based, because these are designed based on the customers usage data. The white box these are guided by the results in the black box, because the black box is black box tests are performed first.

So, if we find that some problems many problems are being detected in some part of the software, we need to do more thorough white box testing on that part. And, that is the reason we can say that the white box testing results are normally guided by the black box testing results.

(Refer Slide Time: 18:07)



Now, let see that this is the past bug data, for some software. Now, let say we are planning to develop some additional parts for this software. Now, how much testing we should do? Let say from the previous one we had seen that reviews get 10 percent of the bugs, unit testing gets about 40 percent of the bugs, integration testing does about 25 percent, system testing gets about 15 percent, and the customer reported is about 10 percent.

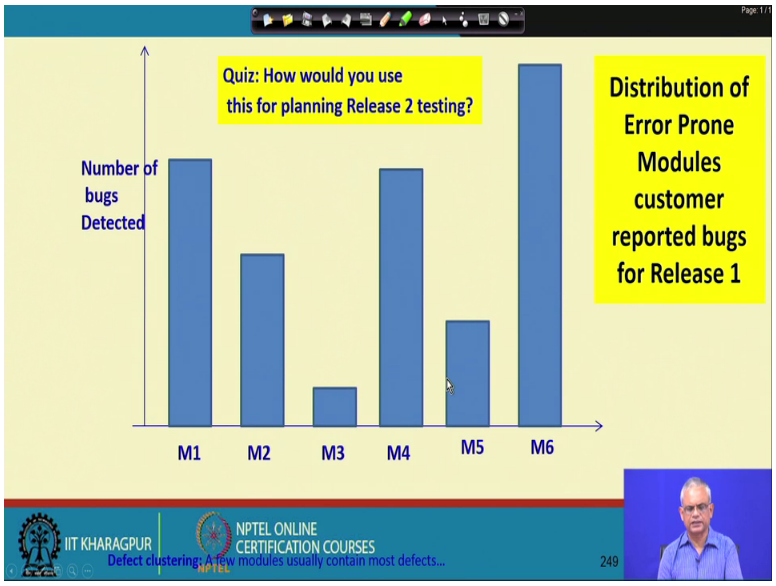
Now, we want to do a new release where we have developed some parts of the software. So, how much let say we want to release it within a week? And, let say total test hours that is available to us, because there are may be 10 testers working may be let say 200 hours of testing. Then for which type of testing review unit testing integration system and customer reported, how much time should be there? How much test time should be allocated for unit testing, for integration testing?

It is common-sense is nothing very profound about this? Since, unit testing is detecting the largest percentage of bugs that is 40 percent; we need to spend more time on unit testing. And to be fair we should spend 40 percent of time, that we have let say 7 days and let say 100 main hours, we should spend 40 percent of that doing unit test, 10 percent of review, 25 percent on integration testing and 15 percent of the system testing.

Now, let say that we have past bug data available to us and we had deployed various types of test strategies, now let say we have return how many bugs we have detected by different test strategies? Let us say strategy one detected 50 percent of the bugs, strategy 230 percent, strategy 310 percent, and twist were customer reported. So, if we want to do a new release for the software, how much time do we plan to do the different types of testing? Test technique one detected 50 percent, test technique 230 percent, test technique 310 percent. Again commonsense, that the test technique one is very effective detecting nearly half of the bug.

And therefore, we must spend 50 percent of the time doing this very thoroughly, 30 percent of the time for test technique 2; test technique 3 is not that effective detected only 10 percent of the problem so, 10 percent of time to that.

(Refer Slide Time: 21:44)

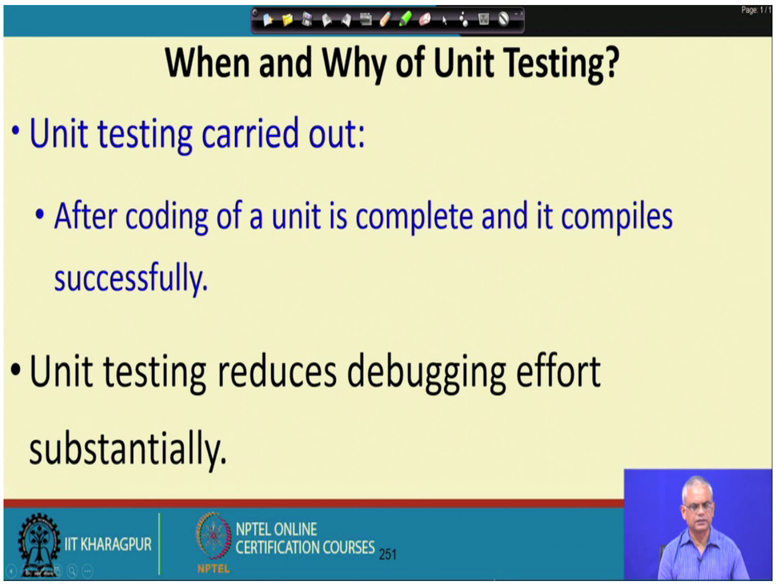

Quiz: How would you use this for planning Release 2 testing?

Distribution of Error Prone Modules customer reported bugs for Release 1

Number of bugs Detected
M1
M2
M3
M4
M5
M6
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
Defect clustering: A few modules usually contain most defects...
249

Now, let say we have another set of data let say based on our unit testing, we found that the module 6 has the maximum number of bugs, module next is module 1 and module 4 and the lowest is module 3. Now, we want to have one more release of this, where we have made changes different parts of the software. So, where do we spend how much time? Again commonsense is that module 6 is the one where maximum numbers of problems are detected so, may be more problems are there. And therefore, we need to spend proportionately much larger time on M 6 and then M 1 M M 4 and M 3 you need to spend the least amount of time.

(Refer Slide Time: 22:52)

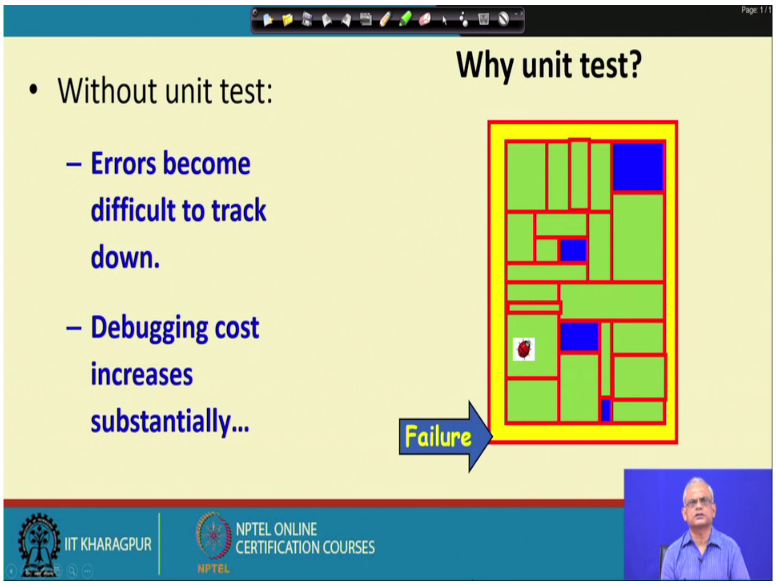


When and Why of Unit Testing?

- Unit testing carried out:
  - After coding of a unit is complete and it compiles successfully.
- Unit testing reduces debugging effort substantially.

Now, let see some details of unit testing. So, far we have been looking at some very basic concepts. Now, let us look at more details on unit testing, but before we start discussing detailed way of designing test cases for unit testing. Let us be clear about one aspect one question, that we might have that can we just eliminate unit testing. And do a thorough integration and system testing.

(Refer Slide Time: 23:44)



Why unit test?

- Without unit test:
  - Errors become difficult to track down.
  - Debugging cost increases substantially…

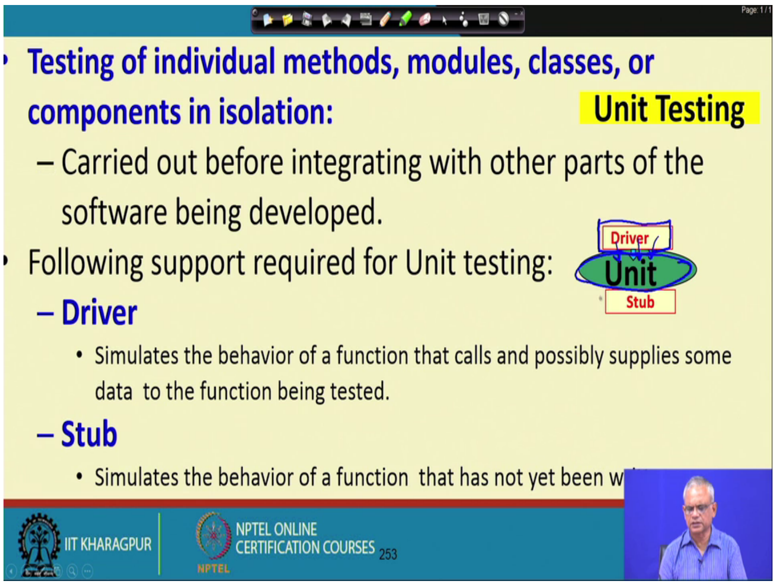
Failure
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES

Unit testing is carried out after coding is complete and, the code has been successfully compiled and that situation we do the unit testing. And, we do not we cannot really eliminate unit testing, because it is a very effective technique, because it reduces the debugging effort. If we do not do unit testing, we do a thorough integration and system testing. Then the debugging effort will be much more, let us understand why? Let us say these are all different units that are shown here, and we just did not do a unit testing and there was let say assume a bug was there.

And, then we run several test cases on some of the test cases they failed, but then to debug we need to check each and every of this unit or may be at least those units, which got executed during the execution of that failed test case. But, in unit testing, when we do unit testing? We know that the bug has to be within that unit, we do not have to look at the other units. And that is the reason, why debugging effort for problem reported during unit testing is much less compare to when unit testing is not done and we directly go for system testing.

In system testing the errors are much more difficult to track down, that is localized which where is the bug. And therefore, the debugging cost increases substantially this is another basic concept in unit testing.

(Refer Slide Time: 25:44)



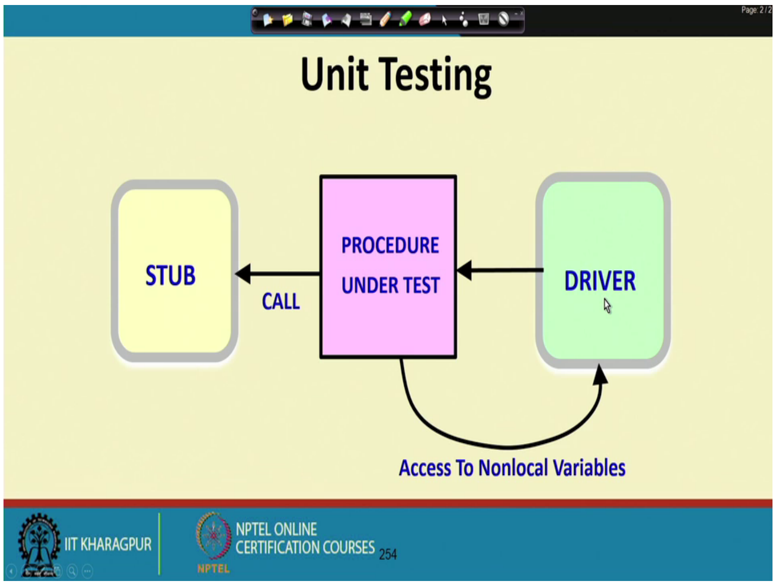
- **Testing of individual methods, modules, classes, or components in isolation:** Unit Testing
  - Carried out before integrating with other parts of the software being developed.
- Following support required for Unit testing:
  - **Driver**
    - Simulates the behavior of a function that calls and possibly supplies some data to the function being tested.
  - **Stub**
    - Simulates the behavior of a function that has not yet been w

We use the term unit testing when we either execute an individual method and test it, unit testing may be applied at the level of modules, we just test unit test a module, we might unit test a class in object oriented code or may unit test a component in a component based software.

But, the problem here is that if we take up a unit and we want to execute it, but then there must be some software which supplies, it data inbox that unit. And, similarly the unit that we are testing it might need to invoke other units there might be call to other functions and so on. And since we are testing the unit in isolation, we need to have this 2 small code written the driver is the one which will call the unit and supplied the necessary data. And the stubs are the functions some dummy functions which the unit will call so, the driver and stub software written during unit testing.

The driver and stub are two, small software so the tester has to write some code here, the driver simulates the behaviour of the function that should call the unit and also supplied the necessary data the test data. They stop on the other hand these are the functions that need to be called by the unit.

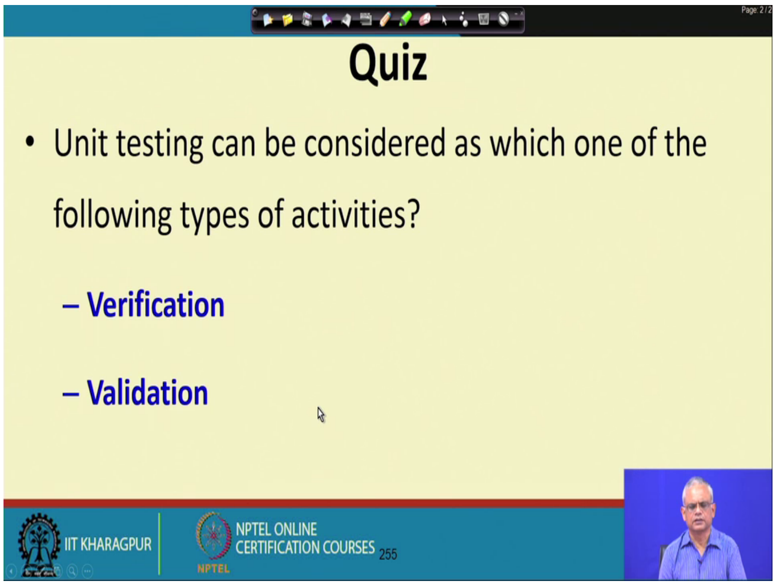
(Refer Slide Time: 27:55)


Unit Testing

STUB
CALL
PROCEDURE UNDER TEST
DRIVER
Access To Nonlocal Variables
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
254

We can represent the same thing in this form, that in order to do unit testing for some unit. The tester has to write 2 small software one is the driver, which calls the unit under test and also supplies it the necessary data and also if the unit needs to excess some global data, then it provides the global variables or global data to the unit. And also needs to write the stub, which the procedure under test that is the unit needs to call. So, for unit testing we need the drivers and stubs to be written before the unit testing can be carried out.

(Refer Slide Time: 28:54)


Quiz

- Unit testing can be considered as which one of the following types of activities?
  - Verification
  - Validation

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
255

Now, let us have a small quiz. So, far we looked at unit testing in this lecture. And we saw that in unit testing we just isolate one unit and then test it, it is adding some test data and we write the driver and stub for that unit, but then unit testing can be considered as which one of the following types of activity is it a verification activity or a validation activity.

The answer is that it is a verification activity, unit testing is a verification activity, because by definition validation is checking the full developed software against it requirements, in unit testing we do not do that we just take out a small unit and then test it against it is functional requirement for that unit. So, unit testing is not validation testing it is a verification testing, we are almost at the end of this lecture we will stop at this point and then continue with the next lecture.

Thank you.